Tutorial Sheet - 2

Q1.)

```
void fun ( int n) {
    int j = 1 , i = 0;
    while (i < n) {
        i = i + j;
        j++ ;
    }
}
```

i	j
0	1
1	2
3	3
6	4
10	5
15	6
⋮	⋮

series = 0, 1, 3, 6, 10, 15 . . . . . .

$$n = 0+1+2+3+ \cdots + k$$

$$n = \frac{k(k+1)}{2}$$

$$n = \frac{k^2+k}{2}$$

$$n \cong k^2$$

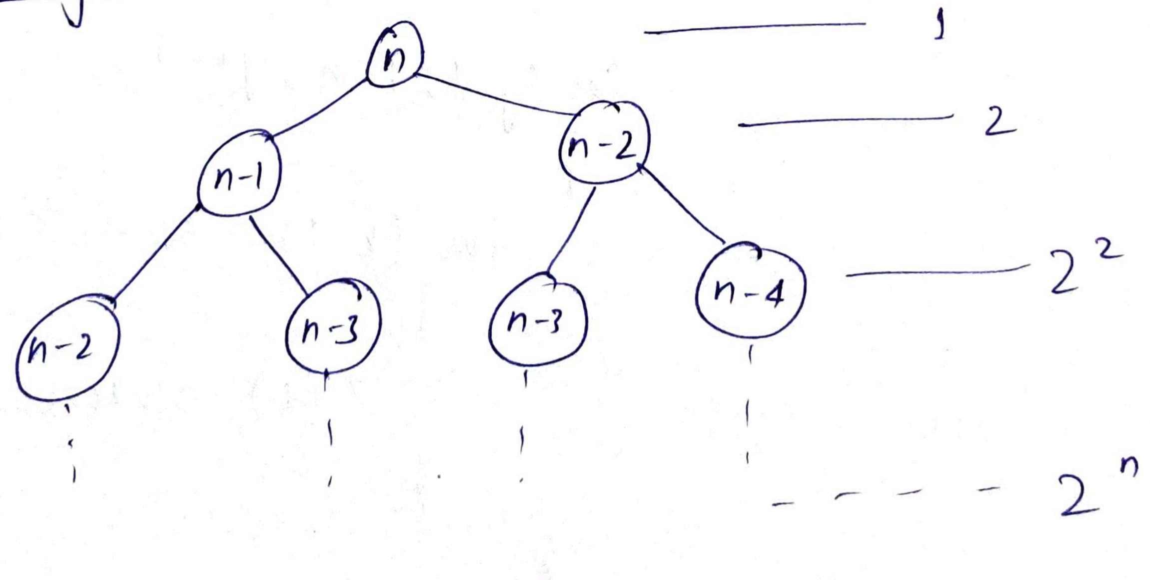
$$k \cong \sqrt{n}$$

Time complexity = $O(\sqrt{n})$

Q2.) Recurrence relation for fibonacci series

$$T(n) = T(n-1) + T(n-2) + 1 .$$

using recursion tree method,



Time complexity $= 1+2+4+ \cdots + 2^n$

$$= \frac{1(2^{n+1}-1)}{2-1} \approx 2^{n+1}-1$$

or Time complexity $= O(2^n)$

* Space Complexity :- Space complexity of fibonacci series using recursion is proportional to height of recurrence tree.

$\therefore$ Space complexity $= O(n)$

Q 3.) Write code for complexity :-

(i) $n \log n$

```
for (i=1; i<=n; i++)
 {
    for (j=1; j<=n; j*=2)
     {
        O(1) statement
     }
 }
```

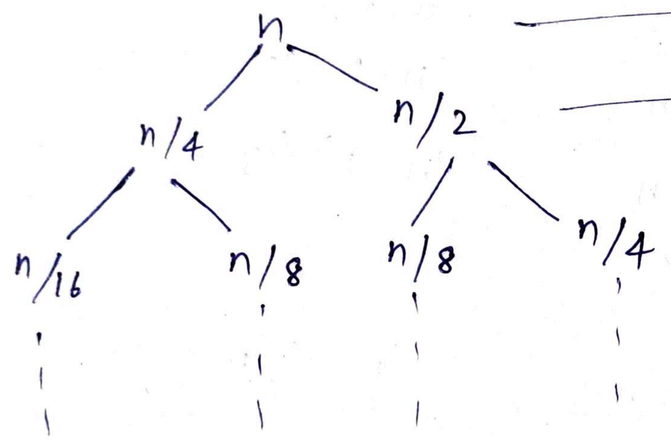
(ii) n^3

```
for (i=1; i<=n; i++)
 {
    for (j=1; j<=n; j++)
     {
        for (k=1; k<=n; k++)
         {
            O(1) statement
         }
     }
 }
```

(iii) $\log(\log n)$

```
int i=n
while (i>0)
{
    i= √i ;
}
```

Q4) $T(n) = T(n/4) + T(n/2) + cn^2$



cn^2

$c\frac{n^2}{16} + c\frac{n^2}{4} = \frac{5cn^2}{16}$

$\frac{cn^2}{256} + \frac{cn^2}{64} + \frac{cn^2}{64} + \frac{cn^2}{16}$

$= \frac{25}{256}cn^2$

so $T(n) = c\left(n^2 + \frac{5n^2}{16} + \frac{25n^2}{256} + \cdots\right)$

here, $r = \frac{5}{16}$ so $S_n = \frac{1}{1-r}$

$T(n) = cn^2\left(1 + \frac{5}{16} + \frac{25}{256} + \cdots\right)$

$= cn^2\left(\frac{1}{1 - 5/16}\right)$

$= cn^2 \cdot \frac{16}{11} = n^2$.

Time complexity = $O(n^2)$.

Q5)

```
int fun(int n) {
    for(int i = 1; i<=n; i++) {
        for (int j=1; j<=n; j+=i) {
            Some O(1) task
        }
    }
}
```

i	j	time
		$(n-1)$
1	1 to n	$(n-1)/2$
2	1 to n	$(n-1)/3$
3	1 to n	
⋮	⋮	
n	1 to n	$(n-1)/n$
		$n \log n$

$\therefore$ Time complexity $= O(n \log n)$

Q6)

```
for (int i=2; i<=n; i=pow (i,k))
{
    // some O(1) expression
}
```

$i = 2, 2^k, 2^{k^2}, 2^{k^3}, \ldots\ldots 2^{k^x}$

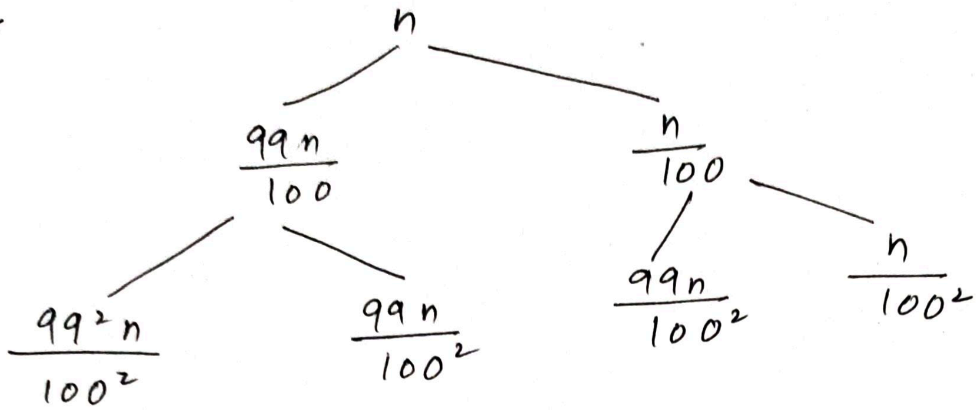
$n = 2^{k^x}$

$\log n = k^x \log 2$

$\frac{\log . \log n}{\log 2} = x \log k$

$x = \frac{\log \log n}{\log 2 * \log k}$. $\boxed{T.C = O(\log . \log n)}$

Q.7)



Taking longer branch that is $\frac{99n}{100}$

Time complexity $= \log_{\frac{100}{99}} n$

$\cong \log n$

$n = \left(\frac{99}{100}\right)^k$

or $k = \log\left(\frac{100n}{99}\right)$

$T(n) = n\left(\log \frac{100}{99}\right)^n / 100$

$= O(n \log_{99} n)$

Q8.) Increasing order of rate of growth

(a) n, $n!$, $\log n$, $\log\log n$, root(n), $\log(n!)$, $n\log n$, $\log^2 n$, 2^n, 2^{2^n}, 4^n, n^2, 100

$100 < \log\log n < \log n < \sqrt{n}$ (root(n)) $< n < n\log n < n^2 < 2^n < 2^{2^n} < 4^n < n!$

(b)

$1 < \log\log n < \sqrt{\log(n)} < \log n < \log 2n < \log n < n < 2n < 4n < n\log n < n^2 < \log(n!) < 2^{2^n} < n!$

(c) $96 < \log_8 n < \log_2 n < 5n < n\log_8(n) < n\log_2 n < 8n^2 < 7n^3 < \log n! < 8^{2n} < n!$